

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application for:

**Embedded Language Interpretation for Configuration of
Fixturing Applications**

Inventors: Terrence Jones
Gregory Brandes

Docket Number: 10010587-1

1 **Embedded Language Interpretation for Configuration of Fixturing**
2 **Applications**

5 **TECHNICAL FIELD**

7 The present invention relates generally to test and evaluation equipment and
8 more specifically to the use of a fixture system in automated or semi-
9 automated environments.

11 **BACKGROUND OF THE INVENTION**

13 A fixturing device is a mechanical device that is capable of holding a product
14 stationary and supports motions that allow for connection to the product. A
15 fixture system comprises a fixturing device, and the associated hardware and
16 software that are used to interact with the product under test. A fixture system
17 is often used in an automated or semi-automated environment. The product
18 that resides within a fixturing device can require customization for integration
19 between the fixturing device and the product. Some examples of products
20 that may be placed within a fixturing device of a fixture system are cellular
21 phones, printed circuit boards, portable receivers, etc.

23 There are several approaches that are currently used to control the manner in
24 which the fixturing device interacts with the product and the environment
25 external to the fixturing device. Referring to the block diagram 100 of FIG. 1,
26 a first type of fixture system is shown. The fixturing device within this fixture
27 system does not utilize an embedded controller or firmware. Thus, all control
28 of the fixturing device is accomplished through cables to an external

1 controller. This type of fixturing device depends upon external software for its
2 basic operation, and is not able to operate autonomously. Fixturing device
3 110 is polled by control software 130. Supervising test software 120 receives
4 the results of the polls executed by control software 130 and sends
5 commands that control software 130 uses to change the state of fixturing
6 device 110. As the software responsible for testing the device under test
7 (DUT), test software 120 must interact to varying degrees with the fixture.

8
9 In **FIG. 1**, the responsibility for correct fixturing operation resides external to
10 the fixturing device of the fixture system. It is also possible to place the
11 capabilities for controlling the fixturing device within the physical confines of
12 the fixturing device. This possibility is shown in the block diagram 200 of
13 **FIG. 2**. This type of fixturing device 220 has an embedded controller 230 with
14 advanced firmware 240. The firmware 240 supports an extensive command
15 set that includes high-level commands for normal operation. The test
16 software 210 does not need to have intimate knowledge of the fixturing device
17 220 internal operation, although the fixturing device 220 does depend upon
18 the controlling software 210 for basic operation. This is because the
19 controlling software 210 polls the fixturing device 220 for changes in state,
20 prior to executing commands to change the state of the fixturing device 220.
21 Thus, normal operation requires that many commands be transmitted from the
22 controlling software 210 to the fixturing device 220, leading to complexity in
23 the controlling software. Also, changes to the high-level commands
24 supported by the fixturing device 220 require an update to the firmware 240.
25 An example of such a high-level command is "Drawer Open." So, while the
26 controlling software 210 does not require intimate knowledge of the fixturing
27 device's 220 internal operation, the controlling software is preferably capable

1 of handling the monitoring of the fixture state to maintain accuracy of the
2 fixture state of fixturing device 220. It would be advantageous to have an
3 easier approach to updating the high-level functions or commands supported
4 by fixturing device 220.

5
6 A third approach to interfacing with fixturing devices within a fixture system
7 300 is shown in **FIG. 3**. This approach uses an embedded controller 340,
8 such as a programmable logic controller (PLC), embedded within fixturing
9 device 320 to control fixture system operation and to implement logic within
10 the fixture as a state machine. This type of fixturing device is operable to be
11 controlled as a state machine and or from the outside via controlling software
12 310. Often the design using an embedded controller 340 allows the fixturing
13 device 320 to be operated autonomously. The embedded controller 340 is
14 operable to respond to events 330 generated by changes in the hardware
15 state 350 of the fixturing device 320. As a state machine, the fixture will
16 respond to changes in hardware state; in this approach, there are no high-
17 level commands, only states. The external software 310 will trigger the fixture
18 to enter the 'begin test' state (possibly via a command) and will then wait for
19 the fixture to trigger the software that a test is complete. This approach lacks
20 the flexibility of easy modification and the capability of being command driven
21 as well.

22
23 There is an unmet need in the art for a fixturing system comprising a fixturing
24 device that may be easily programmed, operate autonomously, be easily
25 upgradeable, and limit the amount of communication between the fixturing
26 device and the controlling software of the fixture system.

1
2 **SUMMARY OF THE INVENTION**

3
4 It is an object of the invention to allow the fixturing device to operate without
5 external interaction (autonomously).
6

7 It is a further object of the invention to limit the number of commands sent
8 from the controlling software to software internal to the fixturing device.
9

10 It is yet another object of the invention to enable time critical commands to be
11 executed without a dependency on the controlling software.
12

13 It is yet another object of the invention that the fixturing device be
14 upgradeable independent of the changes to the firmware.
15

16 It is a further object of the invention that macros be executed by stimulus
17 internal to the fixture.
18

19 It is yet another object of the invention that macros be available upon
20 powering up the fixturing device without the need to download from controlling
21 software.
22

23 Therefore, according to the structure and method of the present invention, an
24 embedded language interpreter for the configuration of fixturing devices is
25 disclosed. The structure of the present invention utilizes firmware internal to
26 the fixturing device to control the internal fixturing device state autonomously.
27 The firmware interacts with macros stored in memory. In the preferred

1 embodiment of the present invention, the macros are stored in local
2 nonvolatile memory. These macros are triggered by events generated by
3 changes in the state of the fixturing device hardware. The firmware is also
4 operable to interact with external controlling software, in order to make
5 changes to the firmware, macros, or the state of the fixturing device hardware.
6 The method of the present invention allows the behavior of a fixturing device
7 to be quickly and programmatically modified without the need to change the
8 fixturing device's firmware or controlling software of the fixture system. The
9 method employs a custom macro language that is used to describe the
10 operation of the fixturing device. The firmware monitors particular hardware
11 states as referenced by an event. When a hardware state change matches
12 what is specified by the event, a specified macro is triggered to execute.
13 Events are programmable and stored in non-volatile memory just like macros.
14 They differ in that they are not interpreted, but they are programmed along
15 with all of the other macros. The use of pre-stored events allows the fixturing
16 device firmware to autonomously trigger the execution of a pre-stored macro
17 and therefore run without the need for controlling software. The individual
18 custom macros are interpreted at runtime, therefore allowing changes to the
19 operation of the fixturing device without changes being made to the fixture
20 device's firmware or the controlling software. The macro language created for
21 a specific application can be very specialized to a fixturing device type. This
22 allows for macro creation and modification by individuals not trained in the
23 internal workings of the software, the fixture system and the fixturing device.

24

1

2 **BRIEF DESCRIPTION OF THE DRAWINGS**

3

4 The novel features believed characteristic of the invention are set forth in the
5 claims. The invention itself, however, as well as a preferred mode of use, and
6 further objects and advantages thereof, will best be understood by reference
7 to the following detailed description of an illustrative embodiment when read in
8 conjunction with the accompanying drawings, wherein:

9

10 **FIG. 1** shows a block diagram of a fixture system comprising a fixturing device
11 that is controlled externally, according to the prior art.

12

13 **FIG. 2** shows a block diagram of a fixture system that utilizes advanced
14 firmware and control software to control fixturing device operation, according
15 to the prior art.

16

17 **FIG. 3** shows a block diagram of a fixture system that uses an embedded
18 controller to control fixturing device operation, according to the prior art.

19

20 **FIG. 4** shows a block diagram of a fixture system that utilizes event driven
21 firmware and macros to control fixturing device operation, according to the
22 present invention.

23

24 **FIG. 5** shows a flow diagram for the construction and operation of a fixture
25 system that utilizes advanced firmware, according to the present invention.

26

27

1 **DESCRIPTION OF THE INVENTION**

2
3 The present invention discloses a structure and method for embedded
4 language interpretation for the configuration of fixturing device applications.
5 Referring to **FIG. 4**, the structure of a fixture system 400 according to a
6 preferred embodiment of the present invention is shown. Controlling driver
7 software 420 is coupled to fixturing device 430 via a cable 425; cable 425 may
8 be an RS-232 cable. Fixturing device 430 is a mechanical device that is
9 operable to support a product and further supports motions that allow for
10 connection to external devices. Fixturing device 430 contains electronic
11 resources such as sensors, switches, LEDs, multiplexors, and relays.
12 Fixturing device 430 additionally has an embedded controller 440 having
13 firmware 450 capable of processing events, external commands and macro
14 via event processor 452, external command processor 454, and macro
15 processor 456 functionality, respectively. Controlling software 420 interacts
16 with fixturing device 430 through one or more commands that are sent to
17 fixturing device 430 via cable 425.

18
19 Automation test software 410, which is coupled to controlling software 420,
20 monitors the state of the controlling software and responds to hardware state
21 errors. Test software 410 monitors fixture state 460 (via the control software
22 420), looking for the state "ready to test". This state indicates to the test
23 software 410 that it can begin testing the DUT. Fixturing device 430 acts
24 autonomously without direct control with respect to low-level controls, such as
25 "Close Drawer," "Engage DUT,"..., "Lower Speaker," interacting with the
26 fixture's operator, and is not dependent on the test or controlling software 410,
27 420. When the test software 410 completes a test, it indicates via a command

1 to the controlling software 420 that the test is complete. The controlling
2 software 420 then triggers a macro that opens the fixture to allow the next
3 DUT to be inserted and the cycle repeated.

4
5 Commands sent by controlling software 420 are receivable internal to fixturing
6 device 430 by firmware 450. An additional function of the controlling software
7 420 is to map fixturing device resources to logical names, thus providing a
8 basis for communication between the fixturing device 430 and external
9 software. Macro processor 456 responds to the one or more commands
10 receivable from controlling software 420 by loading and executing one or
11 more macros stored in memory. Other commands from controlling software
12 420 may be standard low-level commands embodied in one or more macros.
13 The macros 450, coupled to firmware 460, contain one or more executable
14 statements that have been previously compiled by controlling software 420.
15 Macros 460 may be triggered by controlling software 420 or by one or more
16 events of fixturing device 430. Events are triggered by a change in the
17 hardware state 470 of fixturing device 430 and are processed by event
18 processor 452. Events 440 are coupled to the one or more macros and to the
19 hardware state 470 of fixturing device 430.

20
21 According to the method of the present invention, the behavior of fixturing
22 device 430 may be quickly and programmatically modified without making
23 changes to the firmware 460 or the controlling software 420. The method
24 uses a custom macro language that specifies the operation of the fixturing
25 device 430. Events 440 are used to trigger the execution of a pre-stored
26 macro and are operable to run without supporting software. Referring now to
27 FIG. 5, a high level flow diagram 500 according to an embodiment of the

1 method of the present invention is shown. A user of the invention first writes
2 high-level source code (block 510). The macros may be written as source
3 code in a standard text file and compiled by the controlling software and
4 downloaded to the fixture at blocks 520 and 530. In the preferred
5 embodiment, the source code is compiled into byte-code by the control
6 software 420. The byte-code is then transferred by the external control
7 software 420 to firmware 450. Firmware 450 selects the location in
8 nonvolatile memory for the compiled macro byte-code.

9
10 Events are programmed along with the macros in the macro source code file.
11 When the controlling software 420 compiles the macros it also compiles the
12 events and sends both to the firmware 450. Events are different from macros
13 in that they are not interpreted by the firmware; instead they are compared by
14 the firmware against current hardware state changes looking for matches.

15
16 One of the functions of the control software 420 is to associate one or more
17 events with one or more macros (block 540). At the completion of this
18 association, events are placed in non-volatile memory (block 550). Prior to
19 use of the fixturing device 430, the control software 420 is operable to
20 compare the macro revisions stored in the fixture nonvolatile memory,
21 representing the macros currently stored in memory, with the revision
22 specified by the default macro file. If the revisions do not match, control
23 software 420 downloads compiled macros to nonvolatile memory from a
24 default file. The fixturing device may now be used with the firmware and
25 controlling software. It should be noted that it is possible for macros to be
26 triggered by stimuli, including internal events and external events (block 560).
27 An example of an internal event is a change in the hardware state of the

1 fixturing device 430. An example of an external event is a command sent by
 2 the control software 420, and receivable by the macro processor firmware 456
 3 that causes a macro to be executed. When a macro is to be executed, the
 4 firmware 456 copies the byte-code from non-volatile memory into RAM. The
 5 firmware 460 then interprets the byte-code (block 570). The firmware 460
 6 operates on the byte-code by executing each command as it is interpreted
 7 (block 580). In the preferred embodiment of the present invention, the byte
 8 code is interpreted using a recursive command-interpreting algorithm.
 9 Sample macro pseudocode for programming an event is shown below. While
 10 the macro code is written in a custom language, it is similar to other simple
 11 high-level programming languages. Please note that text contained within the
 12 markings `/* ... */` represent what would be comments in the macro source
 13 code.
 14
 15 `/* This line of source code causes an event association to be set. Specifically, it`
 16 `associates Event #3 with a change in the state of a hardware switch. When this switch`
 17 `is toggled, macro #23 will be executed */`
 18 `#Event(3, 23, 4, 1, 1, 20);`
 19
 20 `/* MacroSet indicates that the following is the detailed description of a particular`
 21 `macro number */`
 22 `MacroSet(139)`
 23 `{`
 24 `/* These lines show the use of conditional operators used to check the state of`
 25 `some of the fixture's hardware sensors */`
 26 `(VerifySensorLow(sensorDut1Dis) ?`
 27 `(SensorRead(sensorDut1Eng) ? Return(0) : Continue()) : Continue());`

```

1
2      /* Setting a variable (register) value */
3      RegisterWrite(1, 0);
4
5      /* Writing the state of a hardware valve */
6      ValveWrite(valve03_Dut1, 1);
7
8      /* Labels are used for branching (looping) See: Jump(1) */
9      Label(1);
10
11     /* Increment a variables value */
12     RegisterIncrement(1);
13
14
15     /* Determine the macros flow conditional on hardware sensor state */
16     (VerifySensorLow(sensorDut1Dis)
17       (SensorRead(sensorDut1Eng) ? Return(0) : Continue() ) : Continue());
18
19     /* Use of the language's comparison operation to determine flow */
20     ( (RegisterRead(1) == 3) ? Jump(9) : Continue() );
21
22     /* Cause a delay in the execution of the macro */
23     Delay(2);
24     /* Branch to the label #1. See: Label(1)
25     Jump(1);
26
27     Label(8);

```

```

1
2      /* Cause macro to post an error indicating failure. This error code will be seen
3  by software outside of the firmware */
4      PostError(0xF0, 1);
5
6      /* End the execution of the macro */
7      Return(1);
8
9  Label(9);
10
11     /* Change the state of a hardware valve */
12     ValveWrite(valve03_Dut1, 0);
13
14     /* Post an error indicating failure */
15     PostError(0xF0, 71);
16
17     /* End the execution of the macro */
18     Return(1);
19 }
20

```

21 In the preferred embodiment of the present invention, the firmware is located
22 on a chip with RAM. For example, the firmware could be located on a
23 Motorola 68HC912D60 chip, with 2k of RAM. Also in the preferred
24 embodiment, the fixturing device 430 contains nonvolatile memory. For
25 example, 8k of nonvolatile memory, with an additional 8k of nonvolatile
26 memory on a removable device coupled to the fixturing device 430.

27

1 In light of the foregoing, it is clear that the fixturing device of the present
2 invention is operable to respond to events that are generated internally, said
3 events causing one or more macros to be executed by firmware internal to the
4 fixturing device. The structure of the fixture system also allows the number of
5 commands sent from the controlling software to the internal software of the
6 fixturing device to be reduced. This reduction further allows time critical
7 commands to be executed without dependency on the controlling software.
8 The choice of a macro language that is indirectly executed in response to
9 stimuli allows the fixturing device be upgradeable independent of the changes
10 to the firmware. Said macros are operable to be available upon powering up
11 the fixturing device without the need to download from controlling software.

12
13 While the invention has been particularly shown and described with reference
14 to a preferred embodiment, it will be understood by those skilled in the art that
15 various changes in form and detail may be made therein without departing
16 from the spirit and scope of the invention.

17